ELSEVIER

Journal of Computational and Applied Mathematics 51 (1994) 327–338

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

A quasi-Newton method using a nonquadratic model

Adel F. Saadallah *

Department of Mathematics, Faculty of Science, Helwan University, Cairo, Egypt

Received 11 February 1992; revised 25 September 1992

Abstract

Most "quasi-Newton" methods in common use for function minimisation use a quadratic model to approximate the underlying objective function at the current estimate of the minimum. In this paper we describe new minimisation algorithms derived by replacing the quadratic model with a polynomial type model involving a free parameter which is determined implicitly by means of information about the objective function. We show how this information may be efficiently utilised in an optimisation method of quasi-Newton type. Numerical tests were carried out to demonstrate the performance of the new algorithms in comparison to the standard BFGS method.

Keywords: Unconstrained optimisation; Quasi-Newton methods

1. Introduction

The problem of finding an unconstrained local minimiser $x^* \in \mathbb{R}^n$ of a twice continuously differentiable function $f(x)$, with $x \in \mathbb{R}^n$, may be dealt with using quasi-Newton methods. The philosophy underlying these methods is to build a convex quadratic model for the function f at the current estimate of the minimum x_j

$$F(x) = f(x_j) + g_j^{\mathrm{T}}(x - x_j) + \tfrac{1}{2}(x - x_j)^{\mathrm{T}} B_j (x - x_j), \tag{1}$$

where $g_j = g(x_j)$ is the gradient of f, B_j is a positive definite matrix which is an estimate of $G(x_j)$ (the Hessian of f). Different quasi-Newton methods are distinguished principally by the manner in which they construct the approximations to the Hessian B (or its inverse H). Such methods include the DFP formula (Davidon [4], Fletcher and Powell [8]), the BFGS formula (Broyden [3], Fletcher [7], Goldfarb [11], Shanno [17]) and the so-called "self-scaling variable

* Correspondence to: 29 Amin El-Khuly street, Apt. 4, Ismaelia Square, 11351 west Heliopolis, Cairo, Egypt.

SSDI 0377-0427(92)00115-6

metric" methods (SSVM) [13,14]. To locate the next estimate x_{j+1}, we determine p to minimise the quadratic model (1), which gives

$$p_j = -B_j^{-1} g_j; \tag{2}$$

then

$$(x(t_1) =)\ x_{j+1} + t_1 p_j, \quad j = 0, 1, \ldots, \tag{3}$$

where t_1 is a positive scalar (obtained by the line search routine) chosen with the aim of approximately minimising f along the search direction (2). If x_{j+1} does not satisfy the stopping criteria, then B_j is modified according to some updating rule (see, for example, [6])

$$B_{j+1} = U(s_j, y_j, B_j), \tag{4}$$

where

$$(t_1 p_j =)\ s_j = x_{j+1} - x_j, \qquad y_j = g_{j+1} - g_j. \tag{5}$$

The updating matrix B_{j+1} is required to satisfy the "secant" equation [6]

$$B_{j+1} s_j = y_j, \tag{6}$$

which is regarded as an approximation to the "Newton equation" [9]

$$G(x_{j+1})s = t_1 \left.\frac{\mathrm{d}g(t)}{\mathrm{d}t}\right|_{t=t_1}. \tag{7}$$

The advantage of this approach is that since, near its minimum, any nonlinear function is approximately quadratic, local convergence behaviour for such a function is likely to be similar to that for the quadratic situation. However, such methods may be criticized on the ground that information about the objective function, such as the curvature and the function values, does not influence the updating process as indicated by (4).

Spedicato [19] derived a minimisation algorithm based on nonlinear scaling over the objective function of the form

$$F = F[q(x)], \quad \frac{\mathrm{d}F}{\mathrm{d}q} > 0, \text{ for } x \neq x^*, \tag{8}$$

where q is assumed to be a convex quadratic. He showed that a certain property of invariancy could hold if y_j, in (4), is replaced by

$$y_j = \frac{g_{j+1}}{(\mathrm{d}F/\mathrm{d}q)_{j+1}} - \frac{g_j}{(\mathrm{d}F/\mathrm{d}q)_j}. \tag{9}$$

The determination of the derivative $\mathrm{d}F/\mathrm{d}q$ requires that the function model F is explicitly available. In particular, Spedicato considered three models, namely,

(a) $F = rq(x)$, for some constant $r > 0$,

(b) $F = -\exp(-q)$,

(c) $F = q^r/2r$, for some constant $r > 0$.

Davidon [5] has proposed a new class of algorithms based upon replacing the quadratic model with a conic function model. Ford and Saadallah [9] used the rational model involving a

free parameter. Saadallah [15] described a new algorithm based on the polynomial function model (which we will describe in the next section) involving a free parameter. This parameter is determined implicitly by means of the information contained in the current approximate Hessian.

In the following, we construct new algorithms based upon replacing the quadratic model with a polynomial function model where the free parameter is determined implicitly by employing estimates of the curvature of the objective function at the points x_j and x_{j+1}. We shall, from now on, omit the use of the subscript j and replace the subscript $j+1$ by the subscript 1.

2. A polynomial function model

We approximate the underlying objective function f, at the current point, by a polynomial model

$$F = q(1 + \theta q), \tag{10}$$

where θ is a parameter,

$$q = q(x) = \tfrac{1}{2} e^{\mathrm{T}} A e \tag{11}$$

is quadratic (called "the associated quadratic") with positive definite symmetric matrix A, and

$$e = x - x^*. \tag{12}$$

Calculating the gradient and the Hessian of F from (10), we obtain

$$\nabla F = m \nabla q, \tag{13}$$

$$\nabla^2 F = m \nabla^2 q + 2\theta \nabla q \nabla q^{\mathrm{T}}, \tag{14}$$

where

$$m = 1 + 2\theta q, \tag{15}$$

so that F, like q, has a stationary point at x^*, and since $q(x^*) = 0$, F has a minimum at x^*; from (14), $\nabla^2 F = A = \nabla^2 q$.

From (13) the gradient of F (regarded as a function of t) in a chosen search direction p is

$$g(t) = g(x + tp) = m(t) \nabla q(t), \tag{16}$$

where

$$q(t) = q(x + tp) = \tfrac{1}{2} e(t)^{\mathrm{T}} A e(t), \tag{17}$$

$$e(t) = x + tp - x^* = e_0 + tp. \tag{18}$$

Then, we may write

$$g(t) = m(t)(Ae_0 + tAp) \tag{19}$$

and

$$m(t) = 1 + 2\theta\left(q_0 + tp^{\mathrm{T}} A e_0 + \tfrac{1}{2} t^2 p^{\mathrm{T}} A p\right). \tag{20}$$

At this point, it is appropriate to introduce some further notation:

$$\gamma_0 = p^T g, \quad \gamma_1 = p^T g_1 \quad \text{and} \quad \gamma = \frac{\gamma_1}{\gamma_0}, \tag{21}$$

$$F_1 = F(x_1), \quad F_0 = F(x) \quad \text{and} \quad \delta = p^T A p, \tag{22}$$

$$\rho = \frac{2(F_1 - F_0)}{t_1 \gamma_0}, \tag{23}$$

$$m(t_0) = m_0, \quad m(t_1) = m_1, \qquad \text{where } t_0 = 0, \quad t_1 > 0, \tag{24}$$

$$z = \frac{\gamma_0}{m_0} + \tfrac{1}{2} t_1 \delta, \tag{25}$$

$$\mu = \frac{\theta t_1 z}{m_0}. \tag{26}$$

Multiplying (19) by p^T, we have (on substituting $t = 0$)

$$p^T A e_0 = \frac{\gamma_0}{m_0}. \tag{27}$$

Using (20), (25) and (26), we have

$$m_1 = m_0 + 2\theta t_1 z,$$

or

$$\frac{m_1}{m_0} = 1 + 2\mu. \tag{28}$$

Also, from (10), (25) and (26), we obtain

$$F_1 - F_0 = t_1 z(m_0 + \theta t_1 z),$$

or

$$z = \frac{F_1 - F_0}{t_1 m_0 (1 + \mu)}. \tag{29}$$

Note that we do not have access to values of q or ∇q, so that all computations must be arranged so that they depend only on values of F and g $(= \nabla F)$.

3. The new algorithms

Since q is quadratic, from (16), we may model the gradient by

$$g(t) = m(t)(a + bt), \tag{30}$$

where a and b are constant vectors determined from the two points $(t_0, g(t_0))$ and $(t_1, g(t_1))$.

Therefore,

$$a = m_0^{-1} g, \qquad b = \frac{m_1^{-1} g_1 - m_0^{-1} g}{t_1}. \tag{31}$$

In order to derive a suitable approximation to the Newton equation (7), we need to determine

$$w = t_1 \frac{\mathrm{d}g(t)}{\mathrm{d}t}\bigg|_{t=t_1}, \tag{32}$$

which gives

$$w = \left(1 + \frac{t_1 m_1'}{m_1}\right) g_1 - \left(\frac{m_1}{m_0}\right) g. \tag{33}$$

Having obtained the coefficients of g_1 and g in (33), the value of w may be used in any suitable quasi-Newton updating formula (we use the BFGS formula in the numerical experiments, see Section 5) by replacing y with w in (4), which will thus yield a matrix B_1 satisfying

$$B_1 s = w, \tag{34}$$

which is to be regarded as an alternative approximation (in place of the "secant" equation (6)) to the "Newton equation" (7).

Now, using (20), (25) and (26), we have

$$\frac{m_1'}{m_1} = \frac{2\theta z + \theta t_1 \delta}{m_0 + 2\theta t_1 z}, \qquad \frac{t_1 m_1'}{m_1} = \frac{2\mu + [\theta t_1^2 \delta / m_0]}{1 + 2\mu}.$$

Now, since $t_1 \delta = 2z - 2\gamma_0 / m_0$, then

$$\frac{\theta t_1^2 \delta}{m_0} = 2\mu - \frac{2\theta t_1 \gamma_0}{m_0^2},$$

and by using (23), (26) and (29), we have

$$\frac{2\theta t_1 \gamma_0}{m_0^2} = \frac{4\mu(1+\mu)}{\rho}. \tag{35}$$

Thus,

$$\frac{t_1 m_1'}{m_1} = \frac{4\mu\rho - 4\mu - 4\mu^2}{\rho(1+2\mu)}. \tag{36}$$

Hence, from (20) and (36), Eq. (33) becomes

$$w = \left(1 + \frac{4\mu\rho - 4\mu - 4\mu^2}{\rho(1+2\mu)}\right) g_1 - (1+2\mu) g. \tag{37}$$

(We may observe that if $\mu = 0$, then $w = y$.) The key point in making this algorithm practicable is, therefore, the determination of μ. We obtain μ by employing the curvature estimate of the

objective function $F(t) = f(x + tp)$ in the direction p. We use interpolating quadratic polynomials between $t = 0$ and $t = t_1$, based on selection from the four items of data F_0, F_1, γ_0 and γ_1 already available in the direction $x_1 - x$. Using F_0, F_1 and γ_0 yields

$$\text{curvature at } x \simeq \frac{\gamma_0(\rho - 2)}{t_1}. \tag{38}$$

Using F_0, F_1 and γ_1 yields

$$\text{curvature at } x_1 \simeq \frac{\gamma_0(2\gamma - \rho)}{t_1}. \tag{39}$$

Each of these estimates gives rise to a different algorithm.

3.1. Algorithm NQ1

This algorithm based on using w, defined in (37), is used to estimate $G(x_1)s$ where we find μ by matching the curvature obtained from the model (30) and the approximated curvature at the current point given by (38). The curvature of f in the direction p is $p^{\mathrm{T}}\, \mathrm{d}g(t)/\mathrm{d}t$, and, at x ($t = 0$), it has the value (from (30) and (31))

$$\frac{\gamma_0(\gamma m_0/m_1 - 1 + 2\theta t_1\gamma_0/m_0^2)}{t_1}. \tag{40}$$

On equating the two expressions for the curvature given by (38) and (40), we have

$$\rho = \gamma\frac{m_0}{m_1} + \frac{2\theta t_1\gamma_0}{m_0^2} + 1.$$

Then, by using (28) and (35), we obtain

$$8\mu^3 + 12\mu^2 + (4 + 2\rho - 2\rho^2)\mu + \rho(1 + \gamma - \rho) = 0. \tag{41}$$

We choose μ to be the smallest absolute root of (41) because this will cause w to resemble y more closely.

3.2. Algorithm NQ2

In this algorithm we find μ (in (37)) by equating the curvature of f (in the direction p) at x_1 ($t = t_1$) which has the value (from (30))

$$\frac{\gamma_0(\gamma - m_1/m_0 + \gamma t_1 m_1'/m_1)}{t_1}, \tag{42}$$

and the approximated curvature given by (39). This yields

$$\gamma - \rho = t_1\gamma\frac{m_1'}{m_1} - \frac{m_1}{m_0}.$$

Using (28) and (36), we have

$$\gamma - \rho = \frac{4\gamma\mu\rho - 4\gamma\mu - 4\gamma\mu^2}{\rho(1+2\mu)} - 1 + 2\mu.$$

This leads to the quadratic expression

$$4(\gamma+\rho)\mu^2 + (\gamma+\rho)(4-2\rho)\mu + \rho(1+\gamma-\rho) = 0. \tag{43}$$

If we have a real root, we select μ to be the smallest absolute root of (43), otherwise we put $\mu = 0$ (i.e., $w = y$) resulting in a "standard" quasi-Newton method for that iteration.

We have noted that, if F is actually quadratic, then $\rho = 1 + \gamma$ [16]. Hence, for algorithms NQ1 and NQ2, in the region of the minimum (where F is approximately quadratic) $\rho \simeq 1 + \gamma$ and in this case $\mu \simeq 0$, $w \simeq y$ as desired (see (37)).

4. An alternative approach

Since the function model we have adopted is based upon the existence of an underlying strictly convex quadratic function q, an attractive proposal is to consider the construction of an algorithm which directly attacks the problem of minimising q (which, as we have seen, is equivalent to minimising F). From any point x, the step d defined by

$$(\nabla^2 q)d = -\nabla q \tag{44}$$

would yield the exact minimum $x + d$ of q in just one iteration. Using (13) and (14), Eq. (44) becomes

$$\left(\nabla^2 F - \left(\frac{2\theta}{m_0^2}\right) gg^{\mathrm{T}}\right) d = -g. \tag{45}$$

Replacing $\nabla^2 F$ by its approximation B, we therefore obtain

$$Bd = -\left(1 - \left(\frac{2\theta}{m_0^2}\right)\sigma\right) g, \quad \text{where } \sigma = d^{\mathrm{T}} g.$$

It then follows, from (2), that

$$d = \left[1 - \frac{2\theta}{m_0^2}\sigma\right] p, \tag{46}$$

i.e., that p and d are collinear as we expected, since the contour lines of F coincide with the contour lines of q. Pre-multiplying (46) by g^{T} and solving for σ, we find that

$$\sigma = \left[1 + \frac{2\theta\gamma_0}{m_0^2}\right]^{-1} \gamma_0. \tag{47}$$

Table 1

Starting points	BFGS	NQ1	NQ2	NQ1-SCP	NQ2-SCP
Rosenbrock function ($n = 2$)					
(−1.2, 1)	42 (32)	39 (30)	37 (29)*	39 (30)	44 (35)
(−12, 10)	124 (100)	123 (100)	123 (102)	114 (96)*	119 (94)
(6.39, −0.221)	74 (62)	67 (56)*	74 (61)	72 (64)	71 (62)
(−3.635, 5.621)	71 (57)	62 (50)	62 (50)	60 (53)*	64 (53)
Sum	311 (251)	291 (236)	296 (242)	285 (243)	298 (244)
Rosenbrock cubic function ($n = 2$)					
(−1.2, 1)	48 (40)	45 (37)	43 (35)	46 (40)	39 (36)*
(−12, 10)	573 (434)	560 (436)	546 (440)	528 (412)	521 (418)*
(6.39, −0.221)	248 (190)	245 (186)*	247 (200)	245 (192)	251 (209)
(−3.635, 5.621)	128 (102)	111 (92)*	112 (89)	111 (92)*	117 (96)
Sum	997 (766)	961 (751)	948 (764)	930 (736)	928 (759)
Powell badly-scaled function ($n = 2$)					
(0, 1)	203 (161)	192 (160)	200 (166)	187 (149)	186 (160)*
(−1, 5)	201 (159)	189 (156)	194 (166)	181 (158)*	194 (169)
(0.01, 5)	172 (134)	166 (135)	171 (144)	163 (138)*	172 (144)
Sum	576 (454)	547 (451)	565 (476)	531 (445)	552 (473)
Brown badly-scaled function ($n = 2$)					
(1, 1)	61 (27)	48 (25)	53 (27)	50 (32)	42 (25)*
(10, −0.5)	73 (35)	49 (26)	63 (32)	56 (33)	47 (28)*
(−10, 1)	48 (23)	53 (28)	60 (32)	47 (32)*	52 (26)
Sum	182 (85)	150 (79)	176 (91)	153 (97)	141 (79)
Box "difficult" exponential function [2] ($n = 3$)					
(0, −30, 1)	34 (31)	32 (28)	34 (29)	34 (27)	32 (24)*
(−4, 0, 2)	31 (28)	29 (26)*	29 (26)*	29 (26)*	29 (26)*
(−2.66, −3.4, 8)	73 (58)	70 (55)	73 (58)	69 (54)	68 (56)*
Sum	138 (117)	131 (109)	136 (113)	132 (107)	129 (106)
Helical valley function ($n = 3$)					
(−1.2, 1, 1.2)	29 (24)	27 (22)	28 (23)	25 (20)*	26 (22)
(−5, 1, 5)	41 (34)	36 (28)*	36 (31)	39 (32)	44 (35)
(20, 20, 20)	39 (33)	39 (33)	37 (31)	35 (32)	27 (23)*
Sum	109 (91)	102 (83)	101 (85)	99 (84)	97 (80)
Weibull function ($n = 3$)					
(5, 0.15, 2.5)	36 (30)	37 (30)	39 (34)	37 (30)	34 (29)*
(200, 0.1, 40)	85 (64)	72 (51)	70 (53)	72 (54)	68 (51)*
Sum	121 (94)	109 (81)	109 (87)	109 (84)	102 (80)
Wood function ($n = 4$)					
(−3, 0, 3, 1)	65 (56)	65 (54)	60 (53)*	63 (54)	70 (59)
(−1.2, 1, 1.2, 1)	50 (38)	47 (36)	47 (38)	43 (35)*	45 (35)
(−30, −10, −30, −10)	89 (80)	81 (71)	75 (68)*	91 (83)	101 (89)
Sum	204 (174)	193 (161)	182 (159)	197 (172)	216 (183)

Table 1 (continued)

Starting points	BFGS	NQ1	NQ2	NQ1-SCP	NQ2-SCP
Powell singular function ($n = 4$)					
(3, −1, 0, 1)	33 (30)	31 (28)	31 (28)	30 (27)*	32 (29)
(−3, −1, −3, −1)	36 (34)	34 (32)*	37 (35)	34 (32)*	36 (34)
(−1.2, 1, −1.2, 1)	27 (24)	26 (23)	25 (22)*	25 (22)*	25 (22)*
Sum	96 (88)	91 (83)	93 (85)	89 (81)	93 (85)
EXP4 function [1] ($n = 4$)					
(−1, 0, 2, −2)	58 (51)	56 (50)	48 (44)*	50 (45)	49 (44)
(1, 0, −5, 3)	148 (122)	145 (117)	141 (118)*	145 (117)	152 (126)
(−3.562, −3.816, 51.44, −54.06)	233 (172)	215 (164)	220 (175)	210 (167)	208 (173)*
Sum	439 (345)	416 (331)	409 (337)	405 (329)	409 (343)
EXP6 function [1] ($n = 6$)					
(−1, −9, −5, 1, −4, 3)	63 (58)	60 (56)*	60 (57)	62 (58)	61 (59)
(1, −8, −5, 1, 1, 1)	129 (115)	132 (122)	96 (88)*	127 (117)	97 (90)
(0, −5, −3, 3, −5, 3)	89 (84)	79 (75)	68 (65)*	79 (77)	70 (67)
Sum	281 (257)	271 (253)	224 (210)	268 (252)	228 (216)
Extended Rosenbrock function (starting point is (−1.2, 1, −1.2, 1,...))					
$n = 10$	44 (37)	42 (35)*	42 (37)	42 (37)	47 (38)
$n = 16$	43 (37)	43 (37)	45 (36)	41 (35)*	48 (36)
$n = 22$	46 (38)	49 (40)	46 (38)	45 (37)*	47 (37)
Sum	133 (112)	134 (112)	133 (111)	128 (109)	142 (111)
Extended Powell's function (starting point is (−3, −1, 0, 1, −3, −1, 0, 1,...))					
$n = 36$	67 (66)	53 (52)*	61 (60)	64 (63)	57 (55)
Power function [19] (starting point is (1, 1, 1, 1, 1,...))					
$n = 10$	140 (139)	137 (136)	136 (135)	135 (134)	134 (133)*
$n = 20$	281 (280)	279 (278)	276 (275)	275 (274)	273 (272)*
Sum	421 (419)	416 (414)	412 (410)	410 (408)	407 (406)
Grand sum	4075 (3319)	3865 (3196)	3845 (3230)	3800 (3210)	3799 (3220)
Number of best performances	0	9	9	14	15

Using (47), Eq. (46) becomes

$$d = \left[1 + \frac{2\theta\gamma_0}{m_0^2}\right]^{-1} p, \tag{48}$$

from which d may be determined, via (2), provided that the factor $2\theta/m_0^2$ can be suitably estimated. This, however, is not accomplished as easily as might first be expected, since the direction d must be computed before x_1, and thus $2\theta/m_0^2$, can be determined. We solve this difficulty by using information from the previous iteration. Temporarily, let underlined quantities denote quantities related to the previous iteration. We denote the factor $2\theta/m_0^2$ by α.

Then (from (28))

$$\alpha = \frac{2\theta}{m_0^2} = \frac{2\theta}{\underline{m}_1^2} = \frac{2\theta}{\underline{m}_0^2(1+2\underline{\mu})^2},$$

that is,

$$\alpha = \frac{\underline{\alpha}}{(1+2\underline{\mu})^2}. \tag{49}$$

Hence, α and thus d can be determined if $\underline{\alpha}$ and $\underline{\mu}$ have been retained. We have tested two versions of this approach, algorithm NQ1-SCP and algorithm NQ2-SCP. After a line search has been carried out in the direction d, μ can be calculated from (41) for algorithm NQ1-SCP or from (43) for algorithm NQ2-SCP. With this more up-to-date information, it is possible to revise the estimate of α for passing on to the next iteration (from (26) and (29)):

$$\frac{2\theta}{m_0^2} = \frac{4\mu(1+\mu)}{\rho t_1 \gamma_0}.$$

In outline, the structure of an algorithm to implement this approach is as follows.

(1) At the end of an iteration, compute

$$\alpha = \frac{4\mu(1+\mu)}{\rho t_1 \gamma_0 (1+2\mu)^2},$$

to pass on to the next iteration.

(2) At the beginning of the next iteration, determined p (via (2)), and calculate $\gamma_0 = p^{\mathrm{T}} g$.

(3) If $1+\alpha\gamma_0 > 0$, scale p to give $d = (1+\alpha\gamma_0)^{-1} p$, otherwise take $d = p$.

(4) Carry out the line search and determine t_1, x_1, g_1 and μ in the usual fashion; return to step (1).

5. Numerical tests and results

In order to evaluate the proposed algorithms, numerical tests were carried out on several unconstrained optimisation problems. They are listed in Table 1 by their commonly accepted names; (a full description of the functions used may be found in [12]). Each function was minimised from a variety of starting points, giving 40 test-cases in all. For the purposes of comparison, the problems were first solved by the BFGS method, this is achieved by means of the formula

$$B_1 = B - \frac{Bss^{\mathrm{T}}B}{s^{\mathrm{T}}Bs} + \frac{yy^{\mathrm{T}}}{s^{\mathrm{T}}y}. \tag{50}$$

If B is positive definite, then so is B_1, provided that

$$s^{\mathrm{T}}y > 0. \tag{51}$$

This condition guarantees that the vector p, in (2), is a descent direction (see [7]) on every iteration. We employ a cubic interpolation line search technique whenever the initial step in the specified "quasi-Newton" search direction was unacceptable and requiring, for a point x_1 to be acceptable, as the new estimate of the minimum, that the conditions

$$f(x_1) \leqslant f(x) + 10^{-4} s^{\mathrm{T}} g, \tag{52a}$$

$$s^{\mathrm{T}} g_1 > s^{\mathrm{T}} g \tag{52b}$$

be satisfied [18]. The new algorithms were implemented in identical fashion, except that we employ the vector w (Eq. (37)) in place of y in the BFGS updating formula equation (50). Therefore, we require that the stability condition

$$s^{\mathrm{T}} w > 0 \tag{53}$$

be satisfied to produce a positive definite matrix B_1 if B is positive definite. For the new algorithms, if condition (53) is not satisfied, then we take $\mu = 0$ (in (37)) and turn to use $w = y$.

In Table 1, each entry under the algorithm name consists of two numbers: the first gives the number of function and gradient evaluations required by the method for convergence, while the second number (in brackets) denotes the number of iterations required. For each test case, the method which yielded the best performance (determined by number of function/gradient evaluations, with ties resolved on the basis of iterations) is indicated with an asterisk. We scaled H_0 ($= I$, the initial approximation to the inverse Hessian) by $s^{\mathrm{T}} y / y^{\mathrm{T}} y$ for large problems (when $n \geqslant 10$) is recommended in [18].

6. Summary and conclusions

Most quasi-Newton methods in common use for function minimisation use a quadratic model to approximate the underlying objective function. In this paper, we replaced the quadratic model with a polynomial function model involving a free parameter which is determined implicitly by means of information about the objective function. The motivation for this model is to investigate the effect of the particular model on the quasi-Newton methods. The polynomial function model can certainly represent polynomial objective functions better than quadratic functions.

We described new minimisation algorithms based upon the polynomial type model. It has been shown that these algorithms may be expected to degenerate to the standard quasi-Newton method as the minimum is approached. For the algorithms NQ1-SCP and NQ2-SCP, we show that the search vectors generated while minimising F are proportional to the corresponding search vectors for the associated quadratic. The numerical experiments reported in Table 1 indicate that the performance of the new algorithms exhibit a modest, but distinct, advantage over the BFGS method. Overall, the performance of the new algorithms is very similar and it is difficult to choose between them. However, based on the overall results, we would tend to favour NQ2-SCP slightly.

Finally, we conclude, on the basis of the test problems used, that the proposed model behaves as efficiently as the currently used model and may be more satisfactory for the polynomial type problems.

References

[1] M.C. Biggs, A note on minimization algorithms which make use of non-quadratic properties of the objective function, *J. Inst. Math. Appl.* **12** (1973) 337–338.
[2] M.J. Box, A comparison of several current optimization methods and the use of transformations in constrained problems, *Comput. J.* **9** (1966) 67–77.
[3] C.G. Broyden, Quasi-Newton methods and their application to function minimization, *Math. Comp.* **21** (1967) 368–381.
[4] M.C. Davidon, Variable metric method for minimisation, AEC Research and Development Report ANL-5990 (revised), 1959.
[5] M.C. Davidon, Conic approximations and collinear scalings for optimisers, *SIAM J. Numer. Anal.* **12** (1980) 268–281.
[6] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimisation and Nonlinear Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1983).
[7] R. Fletcher, A new approach to variable metric algorithms, *Comput. J.* **13** (1970) 317–322.
[8] R. Fletcher and M.J.D. Powell, A rapidly convergent descent method for minimisation, *Comput. J.* **6** (1963) 163–168.
[9] J.A. Ford and A.F. Saadallah, A rational function model for unconstrained optimisation, in: D. Greenspan and P. Rózsa, Eds., *Numerical Methods,* Colloq. Math. Soc. János Bolyai **50** (North-Holland, Amsterdam, 1988) 539–563.
[10] J.A. Ford and A.F. Saadallah, On the construction of minimisation methods of quasi-Newton type, *J. Comput. Appl. Math.* **20** (1987) 239–246.
[11] D. Goldfarb, A family of variable metric methods derived by variational means, *Math. Comp.* **24** (1970) 23–26.
[12] J.J. More, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software* **7** (1980) 17–41.
[13] S.S. Oren, On the selection of parameters in self-scaling variable metric algorithms, *Math. Programming* **7** (1974) 351–367.
[14] S.S. Oren and E. Spedicato, Optimal conditioning of self-scaling variable metric algorithms, *Math. Programming* **10** (1976) 70–90.
[15] A.F. Saadallah, A new approach to quasi-Newton methods for minimisation, Ph.D. Thesis, Univ. Essex, 1987.
[16] A.F. Saadallah, A rational gradient model for minimization, *J. Comput. Appl. Math.* **39** (3) (1992) 277–286.
[17] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization, *Math. Comp.* **24** (1970) 647–656.
[18] D.F. Shanno and K.H. Phua, Algorithm 500: Minimization of unconstrained multivariate functions, *ACM Trans. Math. Software* **2** (1976) 87–94.
[19] E. Spedicato, A variable metric method for function minimisation derived from invariancy to nonlinear scaling, *J. Optim. Theory Appl.* **20** (1976) 315–329.